REMARKS

Initially, Applicant notes that the amendments and remarks presented herein are consistent with those noted in the recent telephone call from Applicant's representative to the Examiner. Accordingly, entry of this amendment and reconsideration of the pending claims is respectfully requested.

The Office Action, mailed August 20, 2007, considered and rejected claims 1-5, 7-18, 21-28 and 30-38. Claims 1-5, 7-18, 21-28 and 30-38 were rejected under 35 U.S.C. § 103(a) as being unpatentable over *Gombocz* (U.S. Publ. No. 2002/0156792) in view of *Wong* (U.S. Patent No. 6,889,229), and further in view of *Burstein* (U.S. Patent No. 7,076,541).[1]

By this paper, claims 1, 10, 17, 21, 25 and 37 have been amended, while no claims have been added or cancelled.[2] Accordingly, following entry of this amendment, claims 1-5, 7-18, 21-28 and 30-38 are pending, of which claims 1, 10, 17, 21, 25 and 37 are the only independent claims at issue.

As discussed previously with the Examiner, Applicant's claims generally relate to extending the use of data-types between different middle-tier servers in a multi-tier server system, and without transferring the data-type information between middle-tier servers in a peer-to-peer system. As recited in claim 1, for example, a method is disclosed for deploying one or more data types from a back end server of the multi-tier server system to any of a plurality of middle tier servers so as to maintain consistency and compatibility in the definitions of the data types and in the code associated with each data type stored on each of the middle tier servers. As recited, the method can include an act of creating a special table in a database on the back end server, the special table including multiple data elements that each include a data type identification field, a code field with code necessary for enabling use of the data type, and a link to at least one data type specific table with information further defining the data type. The database of the back end server acts as a repository for each data type used by any of the middle tier servers, such that the back end server acts as a single and centralized source from which each of the middle tier servers obtains all data types used by any other of the plurality of middle tier

[1] Although the prior art status of the cited art is not being challenged at this time, Applicant reserves the right to challenge the prior art status of the cited art at any appropriate time, should it arise. Accordingly, any arguments and amendments made herein should not be construed as acquiescing to any prior art status of the cited art.

[2] Support for the claim amendments can be found throughout Applicant's original application, including at least the disclosure in paragraphs 24, 30 and 40 of the originally filed application.

servers as well as the corresponding code for use of the data types. Accordingly, the single and centralized source from which each middle-tier server obtains all data types is at a different tier than the middle tier-servers to which the data type is being extended. An extended assembly that corresponds to the data type to be deployed is also obtained which includes data from the special table such as data identifying the data type, one or more definitions, and code for processing the data. The extended assembly is also transmitted to at least one of the middle tier servers.[3]

While the cited references generally relate to transferring data between computer devices, Applicant respectfully submits that they fail to disclose or suggest, whether cited individually or in combination, each and every limitation of the pending claims. For example, among other things, the cited references fail to disclose or suggest that a back end server maintains a special table which, among other things, links to one or more other data specific tables further defining a corresponding data type, as recited in combination with the other claim elements.

For example, *Gombocz* discloses an object handling system for intelligently handling data according to dynamic application needs. For example, an object translation engine (OTE) is described which connects to multiple external databases and to other optional components. (¶ 193). The OTE is further connected with an application/database generator (ADG) and master query component (MQC) for translating requested data from one type to a different type. (¶¶ 193, 194).

Specifically, the OTE component includes methods for bi-directional interchange with components and access interfaces, and provides methods for processing heterogeneous and incompatible data into recognized and understandable data. (¶ 194). Thus, an application which uses data of a first type can request data and, even if the requested data is of an incompatible data type, the OTE can translate the data to a recognized type. (¶¶ 194, 195). The OTE thus provides methods, protocols and processing components to enable dynamic, automated translation into data types, structures, formats, matrices, and various data content/routing protocols. (¶ 195).

Accordingly, *Gombocz* discloses that an application can use an OTE to obtain data, even though the data is stored in a different format or is of a different type. *Gombocz* fails to disclose,

[3] Independent claims 10, 17, 21, 25 and 37 generally correspond to the method of claim 1. Accordingly, the discussion herein with regard to claim 1 applies equally to each of claims 10, 17, 21, 25 and 37. Applicant notes, however, that claim 37 further recites wherein the middle tier server is an email exchange server, the back end server is an SQL server, the creation of extended assemblies after the SQL server receives a request for a data type, the use of the back end SQL server as the only location of modifications to data types, and that each data element is specific to particular data type and further includes a data name field.

however, that the requesting computer receives the type information, or that a type definition is specified in a special table linking to one or more other tables that are specific to a particular data type and further define the data type as claimed in combination with the other claim elements. Instead, the data is merely obtained by a requesting system in a translated format which is compatible with type information already known to the application. In short, *Gombocz* discloses that data is requested and converted between data types, but fails to disclose that data types are sent from a central repository to multiple middle-tier servers, let alone that data type information of the claimed type is stored or provided to the middle-tier servers.

Wong and *Berstein* are similarly deficient in these regards. More particularly, *Wong* (see Fig. 1) describes a method for peer-to-peer replication of objects between various nodes connected over a network. Disclosed is a replication process for replicating user-defined objects to make them available to other nodes. (Col. 1, ll. 7-10). In *Wong*, users may define classes and generate, store and receive multiple user-defined objects based on each class at a respective computer used by the user. (Col. 1, ll. 39-43). If data is shared with other users on a network, a database may be copied from one node onto a new node that does not have a copy of the database. (Col. 2, ll. 53-56). According to *Wong*, the node on which a user-defined object is located creates a replication group of objects, including any user-defined objects. (Col. 6, ll. 41-56; Col. 8, ll. 25-43). Thereafter, a database server on the node copies data defining the user-defined object to a data structure on a second node. (Col. 9, ll. 25-31). Subsequently, a database server routine replicates the data by first copying the name of the user-defined object from the first node to the new node. (Col. 10, ll. 31-35). Data defining the user-defined object is then copied to the new node, and data defining the database object is then copied from the data dictionary. (Col. 10, ll. 40-45, 52-60). A data dictionary is an object that includes, for example, the name of a particular table, the type of each column in the table, and the name of a user-defined object. (Col. 6, ll. 31-41; Col. 7, ll. 44-46). Finally, the new node may instantiate the database object based on data in the data dictionary, thereby obtaining a substantially identical database object from a peer node.

Thus, reliance on *Wong* by the Office appears to be in error in that *Wong* expressly discloses peer-to-peer replication of objects in a relational database, such that any computer system can replicate its database on any other computer system, as described above. Thus, in contrast to a back end server acting as a single and centralized source from which each of the

middle tier servers obtains all data types used, *Wong* expressly teaches that there is no single source for all data types, as any node can create data types and replicate those data types to any other node. In other words, *Wong* is directly contrary to the pending claims in that it teaches a system in which multiple systems are able to replicate their data to other systems.

Furthermore, *Wong*, while disclosing that a database object is associated with a data dictionary having a name of a user-defined type and replication metadata, fails to disclose or suggest any link in a special table that links a data type definition to an additional data type specific table(s) further defining the data type, as claimed in combination with the other claim elements. This is particularly so considering that the cited references also fail to disclose that each data element includes such a link, along with identification and code fields.

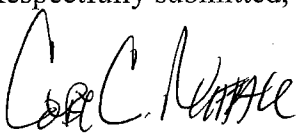
Notably, *Burstein* is no more helpful in these regards. Indeed, *Burstein* is not directed to defining or deploying data types at all, but rather describes a DNS service in which a back-end server accesses shared registry information for a particular domain. (*Abstract*; Col. 6, ll. 25-32). Thus, *Burstein* is silent as to, and has no relevance in, data type replication between servers, let alone a manner in which data type information is stored or transferred between computing systems. Thus, inasmuch as *Burstein* has no information on data types, it also fails to disclose or suggest data elements that each include a data type identification and code field, or a link to data type specific tables further defining a corresponding data type, in the manner claimed.

In view of the foregoing, Applicant respectfully submits that the other rejections to the claims are now moot and do not, therefore, need to be addressed individually at this time. It will be appreciated, however, that this should not be construed as Applicant acquiescing to any of the purported teachings or assertions made in the last action regarding the cited art or the pending application, including any official notice. Instead, Applicant reserves the right to challenge any of the purported teachings or assertions made in the last action at any appropriate time in the future, should the need arise. Furthermore, to the extent that the Examiner has relied on any Official Notice, explicitly or implicitly, Applicant specifically requests that the Examiner provide references supporting the teachings officially noticed, as well as the required explicit statement as to the reason one of ordinary skill in the art would have, at the time of Applicant's invention, modified the art of record in the manner officially noticed.

In the event that the Examiner finds remaining impediment to a prompt allowance of this application that may be clarified through a telephone interview, the Examiner is requested to contact the undersigned attorney at (801) 533-9800.

Dated this 20th day of December, 2007.

Respectfully submitted,



RICK D. NYDEGGER
Registration No. 28,651
JENS C. JENKINS
Registration No. 44,803
COLBY C. NUTTALL
Registration No. 58,146
Attorneys for Applicant
Customer No. 047973

RDN:JCJ:CCN:gd
GD0000002303V001